Scheduling with Quality of Service requirements in Real-Time Energy Harvesting sensors

Maissa Abdallah, Maryline Chetto and Audrey Queudet
IRCCyN
LUNAM University
Nantes, France
e-mail: Firstname.Lastname@irccyn.ec-nantes.fr

***Abstract*—This paper is concerned with the problem of periodic task scheduling in sensor nodes powered by energy harvesters. We address this issue by proposing two energy-aware scheduling algorithms, respectively called Green-RTO and Green-BWP. They aim to guarantee an acceptable Quality of Service (QoS) measured in terms of deadline success ratio.**

***Index Terms*—embedded systems; energy harvesting scheduling; deadlines; energy reservoir; Quality of Service.**

I. INTRODUCTION

Nowadays, embedded systems are in consumer electronics, homes, cars, wearable devices, etc. Consequently, research is very active for designing both very low power embedded platforms and very high energy-density batteries. Nevertheless, the amount of energy that must be available on board still limits their lifespan and prevents their miniaturization. Energy harvesting is a promising approach to solve this problem. Part or all of the operating energy is drawn from ambient energy sources and consequently, the embedded system operates perennially until its hardware failure. Every energy harvesting system must be provided with a power manager that makes the best effort to adapt the power consumption coming from the execution of software processing. Moreover, most of embedded systems are real-time in the sense that a deadline is attached to the execution of every program called task. The goal of a scheduler is then to assign tasks to time slots such that all timing and energy constraints are satisfied. That means that we have to introduce energy-aware scheduling strategies that not only could improve real-time performance but also make a better utilization of ambient energy. In this paper, we focus on energy aware scheduling for uniprocessor energy harvesting systems with strict timing constraints. Our purpose is to gracefully reduce the deadline miss ratio according to specified QoS requirements.

The remainder of the paper is organized as follows. Section II gives background materials on real-time systems. Section III presents energy harvesting technology. Section IV deals with a QoS approach, namely Skip-over. Related works are described in Section V. Section VI describes two novel schedulers which are illustrated in Section VII. Finally Section VIII concludes the paper.

II. BACKGROUND AROUND REAL-TIME COMPUTING

Real-time systems are defined as those systems in which the overall correctness of the system depends on both the functional and the timing correctness. Real-time systems can be classified in three categories: hard, soft and firm [5]. What differenciates them are the degree of tolerance of missed deadlines, usefulness of computed results after missed deadlines, and severity of the penalty incurred for failing to meet deadlines. A firm real-time system must meet its deadlines but with a degree of flexibility. This is in contrast to hard real-time systems where all deadlines have to be met imperatively.

III. ENERGY HARVESTING: WHAT AND WHY?

The goal of any autonomous system such as sensor node is to achieve perpetual functioning without a necessary periodical maintenance due to replacing or recharging battery. Alternative energy sources present in our environment could be exploited to achieve this goal: this is *energy harvesting* (or scavenging). It consists in supplying and converting energy from the surrounding environment and refilling an energy reservoir formed by a battery or by a super-capacitor. Such energy reservoir is required because the embedded system needs to continue operation even when no energy can be drawn, for example at night for a solar-powered system. Possible energy harvesting sources are solar, thermal, vibrational and kinetic energy.

IV. QUALITY OF SERVICE IN FIRM REAL-TIME SYSTEMS

A. Skip-Over: a suitable model

The *Skip-Over model* [4] deals with the problem of scheduling periodic tasks which allow occasional deadline violations (i.e. firm periodic tasks), on a uniprocessor system. A task τ_i is characterized by a worst-case computation time C_i, a period T_i, a relative deadline D_i equal to its period ($D_i = T_i$), and a skip parameter s_i. This parameter represents the tolerance of this task to miss deadlines. That means that the distance between two consecutive skips must be at least s_i periods. When s_i equals to infinity, no skips are allowed and τ_i is a hard periodic task. Every job of a task is either red or blue [4]. A red job must complete before its deadline whereas a blue job can be aborted at any time.

B. Scheduling strategies

Two scheduling algorithms were introduced about fifteen years ago by Koren and Shasha in [4]. The first one is the *Red Tasks Only (RTO)* algorithm. Red jobs are scheduled as soon as possible according to the *Earliest Deadline First (EDF)* algorithm [1], while blue ones are always rejected. The second one is the *Blue When Possible (BWP)* algorithm which is an improvement of RTO. BWP schedules blue jobs whenever their execution does not prevent the red ones from completing within their deadlines. In other words, blue jobs are served in background relatively to red jobs.

C. Problem definition

The performance of the application cannot be predetermined a priori because of the power source that is not entirely predictable and because of possible processing overload. As a consequence, we base the scheduling algorithm on the Skip-over approach in order to process tasks in a best effort manner at runtime and tend to provide the highest QoS (i.e. the least deadline miss ratio).

V. RELATED WORKS

A. The EDeg scheduler

Based on the work presented in [2], El Ghor et al. proposed a real-time scheduling algorithm called *Earliest Deadline with energy guarantee (EDeg)* in [3]. Every task is characterized by an energy consumption in addition to its traditional timing parameters. According to EDeg, the processor executes tasks as soon as possible according to the EDF rule. However, the system starts executing a task only if the so-called *slack energy* is positive and the battery is not empty. Slack energy enables us to quantify the energy consumed by future jobs and to prevent them from violating their deadlines because of energy shortage. Besides the system stops its activity as long as the slack time is positive and the battery is not fully replenished. The key issues in this algorithm are properly predicting the energy production and measuring the current energy level of the battery.

VI. CONTRIBUTIONS: GREEN-RTO AND GREEN-BWP

In this section, we extend the Skip-over approach to the context of energy harvesting applications with real-time constraints. We consider a firm periodic task set τ defined as follows: $\tau = \{\tau_i(C_i, D_i, T_i, s_i, E_i), i = 1...n\}$ where E_i is the Worst Case Energy Consumption (WCEC). Tasks are deeply-red (i.e. synchronous activation at time t=0) and all task deadlines are equal to periods. We define:

$$g_i^*(0, L) = (\left\lfloor \frac{L}{T_i} \right\rfloor - \left\lfloor \frac{L}{T_i.s_i} \right\rfloor)E_i \quad (1)$$

as the energy consumed by red jobs of task τ_i during the interval $[0, L]$. This leads to define the equivalent energy factor U_e^*:

$$U_e^* = max_{L \geq 0} \left\{ \frac{\sum_{i=1}^{n} g_i^*(0, L)}{E(0) + E_r(0, L)} \right\}. \quad (2)$$

E(0) represents the initial level of energy in the battery and $E_r(0, L)$ represents the energy received during the interval [0,L]. In this paper, we assume that the power received by the energy source is constant during an hyperperiod. Then $E_r(0, L) = P_r.L$ where *L* represents the periods' end points of red jobs not beyond the hyperperiod *P* (i.e. $P = LCM(T_1 s_1, .., T_i s_i, .., T_n s_n)$). We deduce the following necessary feasibility conditions: $U_e^* \leq 1$ and $U_p^* \leq 1$ where U_p^* is the equivalent utilization processor defined in [6].

Our approach consists in using the spare time saved by the skipped jobs to recharge the battery whenever necessary as described hereafter.

A. Green-RTO Scheduler

We propose the *Green-RTO scheduler* as a fusion of RTO and EDeg algorithms. EDeg considers hard real-time periodic task sets in the sense that all jobs must complete before their deadlines. Under Green-RTO only red jobs have to be executed before their deadlines.
Green-RTO runs as follows:
- The processor is active if the system has enough slack energy and the battery is not empty. It will execute ready red jobs according to the EDF algorithm.
- The processor is inactive if the slack time is not null or if there is no ready red jobs to be executed.

Slack time at time *t* corresponds to the maximum time available from *t* to postpone red jobs while still satisfying all timing constraints. Its computation uses the *Earliest Deadline as Late as possible (EDL)* algorithm [2] and permits to determine the maximum time interval for recharging the battery while putting the processor to an idle state.

We define the slack energy of a red job J_i of the task τ_i at time *t* as the amount of energy surplus that can be used from *t* by higher priority jobs i.e. jobs with a deadline less than or equal to J_i's deadline d_i.

$$SlackEnergy(t, J_i) = E(t) + \int_t^{d_i} P_r(x)dx - A_i \quad (3)$$

where *E(t)* is the residual capacity at time *t*, $P_r(x)$ is the power of the fluctuating energy source at time *x* and A_i is the total energy required by red jobs ready to be executed after *t* with a deadline less than or equal to d_i.

The slack energy of the system at time *t* will be given by the lowest slack energy of red jobs that should be executed between *t* and *d* (i.e. the deadline of the highest priority job at time *t*).

B. Green-BWP Scheduler

The *Green-BWP scheduler* is based on BWP and Green-RTO algorithms. Blue jobs are executed whenever possible (i.e. when no red job is pending for execution) by taking into account both timing and energy constraints of red jobs. Red jobs are executed in priority according to the EDF rule.

The Green-BWP algorithm has consequently the same framework as Green-RTO and uses the same dynamic data. However, the main differences between Green-RTO and Green-BWP can be summarized as follows:

- Under Green-RTO, the slack time is computed only from the current and future occurring red jobs. Under Green-BWP, the sequence is also constructed upon the execution of current and future occurring red jobs but its computation requires to determine the red jobs' releases according to the current status of blue jobs (i.e. whether they are completed or not). Let us recall that a completed blue job is always followed by a blue one thus introducing a shift (or offset) in the activation pattern of red jobs. In the case of Green-RTO, blue jobs were always considered as uncompleted.
- Under Green-RTO, the *slack energy* is the maximum amount of energy that can be consumed by a red job while still satisfying all constraints of red jobs only. Under Green-BWP, the slack energy at time *t* is the maximum amount of energy that can be consumed by either a red or a blue job while still satisfying all red job constraints. If the job in execution is red at time *t*, the slack energy is computed as with Green-RTO. If the current job in execution at time *t* is blue with deadline d_i, the slack energy of the system represents the minimum between the slack energy of this job and the slack energy of all red jobs with deadline less than or equal to d_i.

VII. ILLUSTRATIVE EXAMPLE

We consider a task set $\tau = \{\tau_i(C_i, D_i, T_i, s_i, E_i)\}$ with $\tau_1(3, 6, 6, 2, 12)$, $\tau_2(4, 10, 10, 2, 12)$ and $\tau_3(6, 15, 15, 2, 18)$.
We give *E(0)* $= E_{max} = 6$ and $P_r = 3$.
$U_p^* = 0,889 < 1$. As $U_p^* < 1$, red jobs are schedulable, abstracting from energy considerations.
$U_e^* = 0,871 < 1$. The equivalent energy utilization is less than the energy received by the battery.

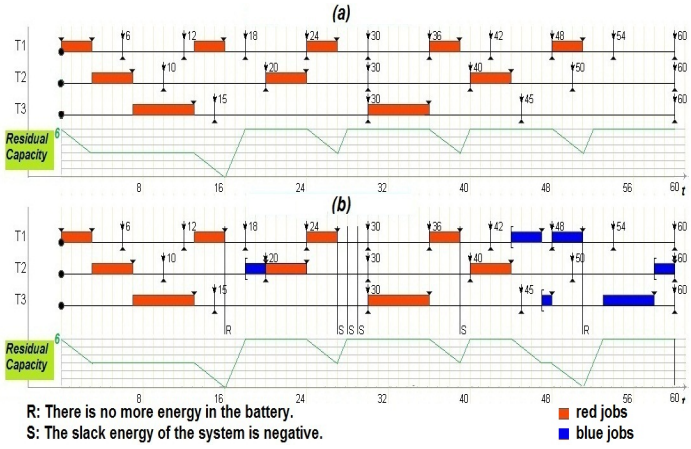

Fig. 1. (a) Green-RTO and (b) Green-BWP scheduling

Figure 1 depicts both the Green-RTO and the Green-BWP scheduling. Under Green-RTO only red jobs are executed and $\frac{1}{s_i}$ job of each task is systematically rejected. Under Green-BWP, all red jobs are executed and some blue jobs are completely executed. It shows that Green-RTO gives a minimale QoS while Green-BWP'one is better. 60% of jobs are completely executed with Green-BWP whereas 50% of jobs are executed with Green-RTO.

Let us comment Figure 1(b): At time $t = 0$, the battery is full. Red jobs are scheduled according to EDF until there is no more energy in the battery (i.e at time $t = 16$). As the residual capacity becomes 0, no job can be processed thus the battery must be recharged. We compute the slack time using the EDL algorithm to put the processor in an idle state and recharge the battery. At time $t = 18$, the battery is fulfilled. As no red job is ready, τ_2's blue job, which has the earliest deadline among all ready blue jobs, is executed. However at time $t = 20$, it misses its deadline. Consequently, τ_2's next job is red. Since τ_2's red job has the earliest deadline among all ready red jobs, it begins its execution as the slack energy of the system is positive and the battery is not empty. At time $t = 24$, τ_1's red job has the earliest deadline and starts its execution as the slack energy of the system is positive. At time $t = 27$, no red job is ready for execution so τ_3's blue job can be processed. However, as the slack energy of the system at time $t = 27$ is negative, the processor remains idle during the time interval given by the slack time computation.

VIII. CONCLUSION AND FUTURE WORKS

Energy harvesting appears as a promising approach to power systems where replacing or recharging batteries is manually impractical. The energy constraint comes to be the main obstacle for increasing the lifespan of autonomous systems such as sensor nodes deployed in hostile surroundings. Energy-aware scheduling becomes an important issue especially in critical real-time applications. The objective is to make the best use of available energy sources and deliver high performance at the same time. In this paper, we presented a real-time energy harvesting model that incorporates energy and timing constraints. We targeted the scheduling problem with periodic tasks with QoS and timing constraints. We proposed two novel scheduling strategies. Several interesting issues need further attention. We would like to incorporate aperiodic tasks in the model as well as task dependencies.

ACKNOWLEDGMENT

The work presented in this paper was partially realized in the framework of the GreenEmbedded project (2011-2012), supported by the CEDRE Programme Hubert Curien.

REFERENCES

[1] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. In *Journal of the ACM, 1973.*
[2] H. Chetto and M. Chetto. Some results of the earliest deadline scheduling algorithm. In *IEEE Transactions on Software Engineering, 1989.*
[3] H. El Ghor, M. Chetto and R. Hajj Chehade, A Real-Time Scheduling Framework for Embedded Systems with Environmental energy Harvesting. In *Computers & Electrical Engineering, 2011.*
[4] G. Koren and D. Shasha. Skip-over algorithms and complexity for overloaded systems that allow skips. In *Proceedings of the 16th IEEE Real-Time Systems Symposium (RTSS'95), 1995.*
[5] J.W.S. Liu. Real-Time Systems, Prentice-Hall, 2000.
[6] M. Caccamo and G. Buttazo, Exploiting Skips In Periodic Tasks For Enhancing Aperiodic Rsponsiveness. In *Proceedings of the 18th IEEE Real-Time Systems Symposium, 1997.*